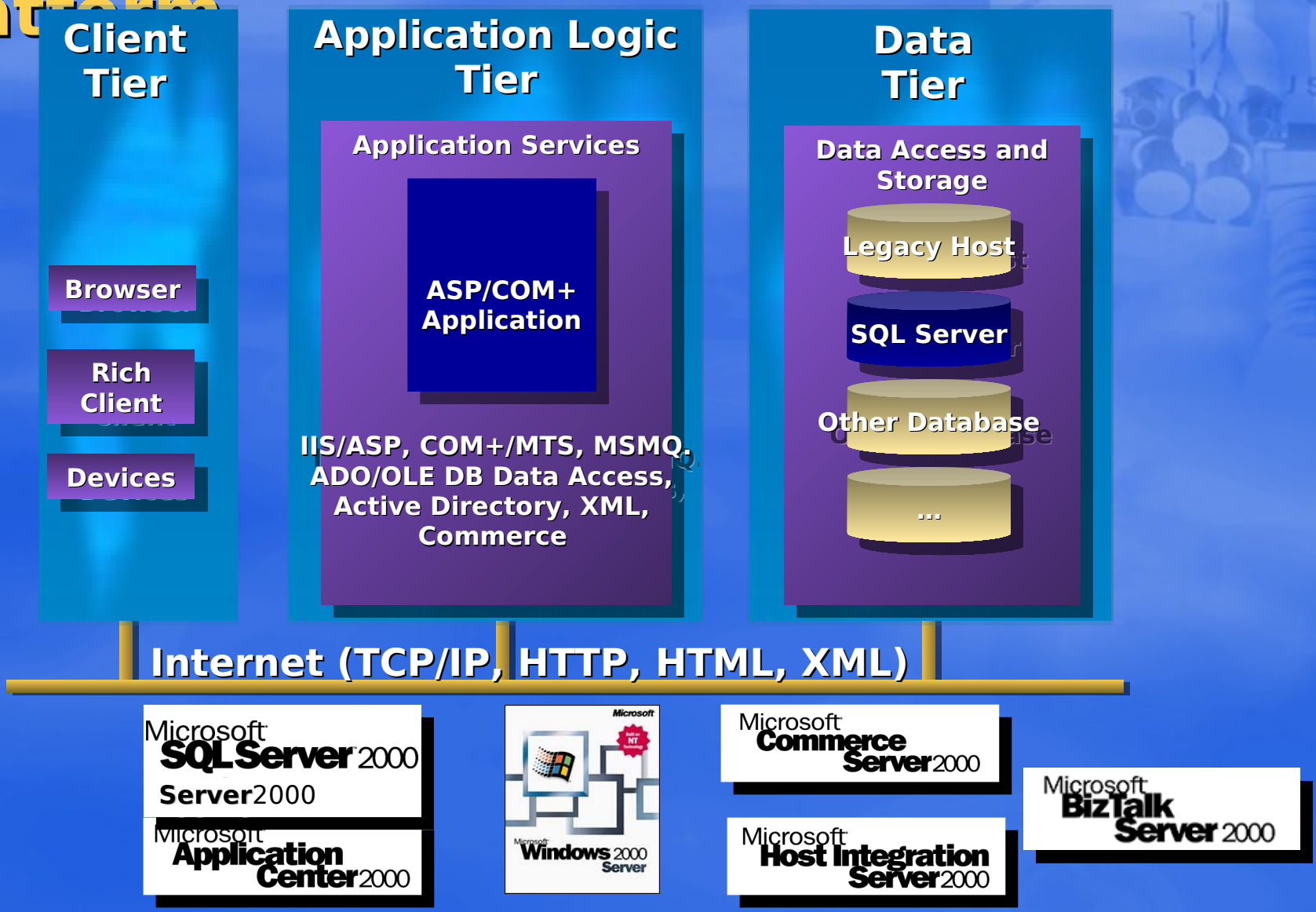
Storage Engine Enhancements

Sameet Agarwal
Development Lead
SQL Server Storage Engine
Microsoft Corporation


POWER UP
Windows DNA 2000
Readiness Conference
featuring SQL Server 2000

Windows DNA 2000

Next Generation Web Application Platform



SQL Server™ 2000 SE Themes

- Performance
- Ease of use
- SAR - ilities
  - Scalability
  - Availability
  - Reliability

Storage Engine Philosophy

- Provide services and base abstractions for server features
- Key attributes are quality, quality, performance, and quality
- Continual code investment maintains key attributes
- It just works…

Storage Engine Roadmap

- 7.0 was a major architectural release
- Rewrote storage engine with state of the art techniques
- 7.0 contains architectural hooks for features we'll add over the next several releases
- Architecture and clean code allows

SQL Server 2000 SE Features

- Performance
- Ease of use
- SAR - ilities
  - Scalability
  - Availability
  - Reliability

Performance - Buffer And I/O Improvements

- Improved parallel I/O scalability
  - Highly optimized I/O scheduling and read-ahead
  - “MAX_ASYNCH_IO” parameter removed. Now uses adaptive feedback algorithm
  - Read-ahead set now includes interior index nodes

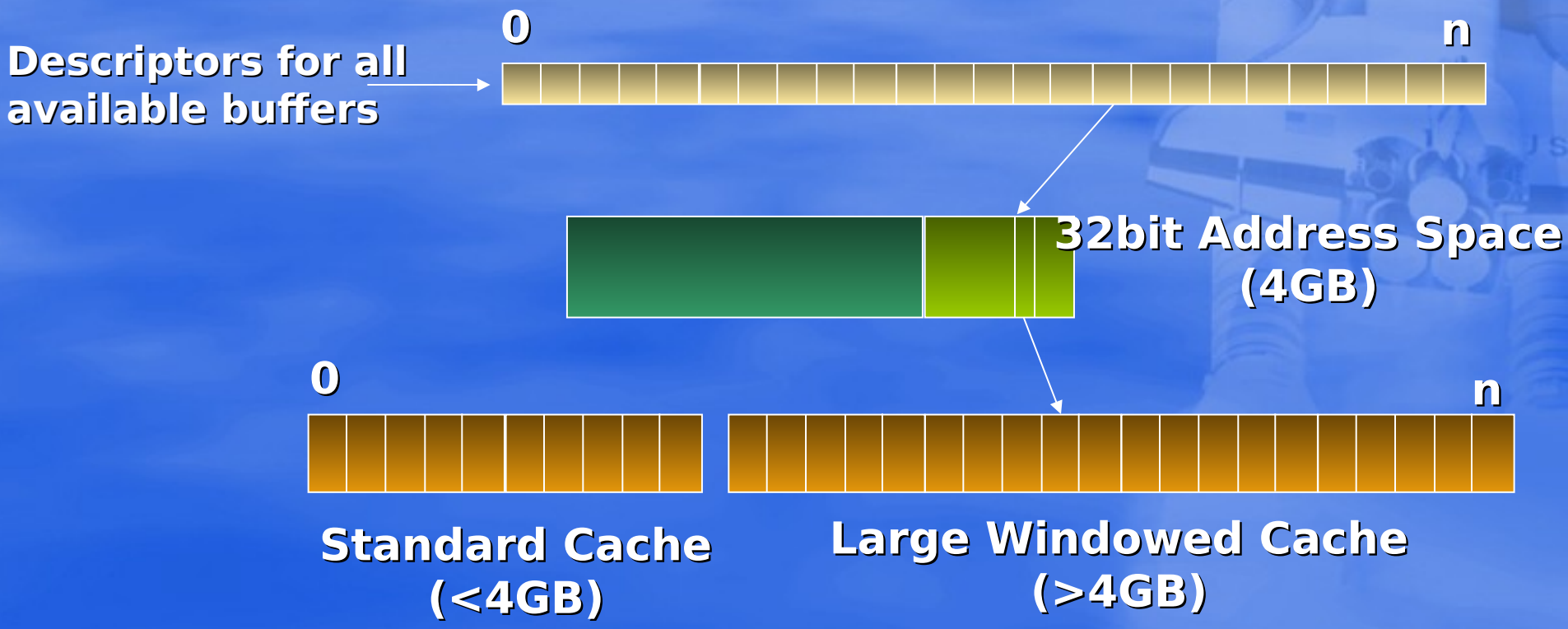
Buffer And I/O Improvements

AWE support (with Windows® 2000)

- Address Windowing Extensions in Windows 2000 allow SQL Server to address up to 64 GB of memory
- SQL Server has unified cache view

  with fast “map”/“unmap” of pages
  to 32-bit pointer

AWE Implementation



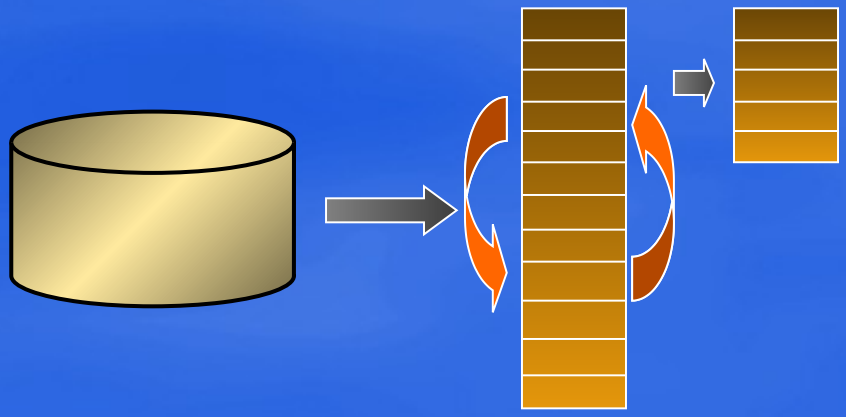
Pages in large windowed cache are “mapped” through a virtual address in the 32-bit process address space to read/write.

Top ‘N’ Sort

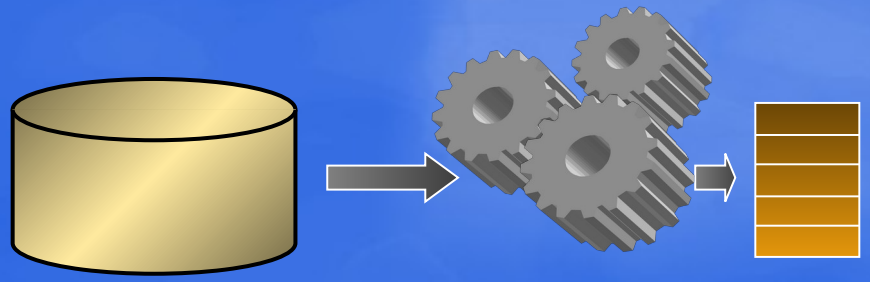
- Efficient Engine to process queries of the form:
  - SELECT TOP 5 * FROM ORDERS ORDER BY ORDER_ID DESC

SQL Server 7.0

SQL Server 2000



Full Sort



Top ‘n’ Engine

SQL Server 2000 SE Features

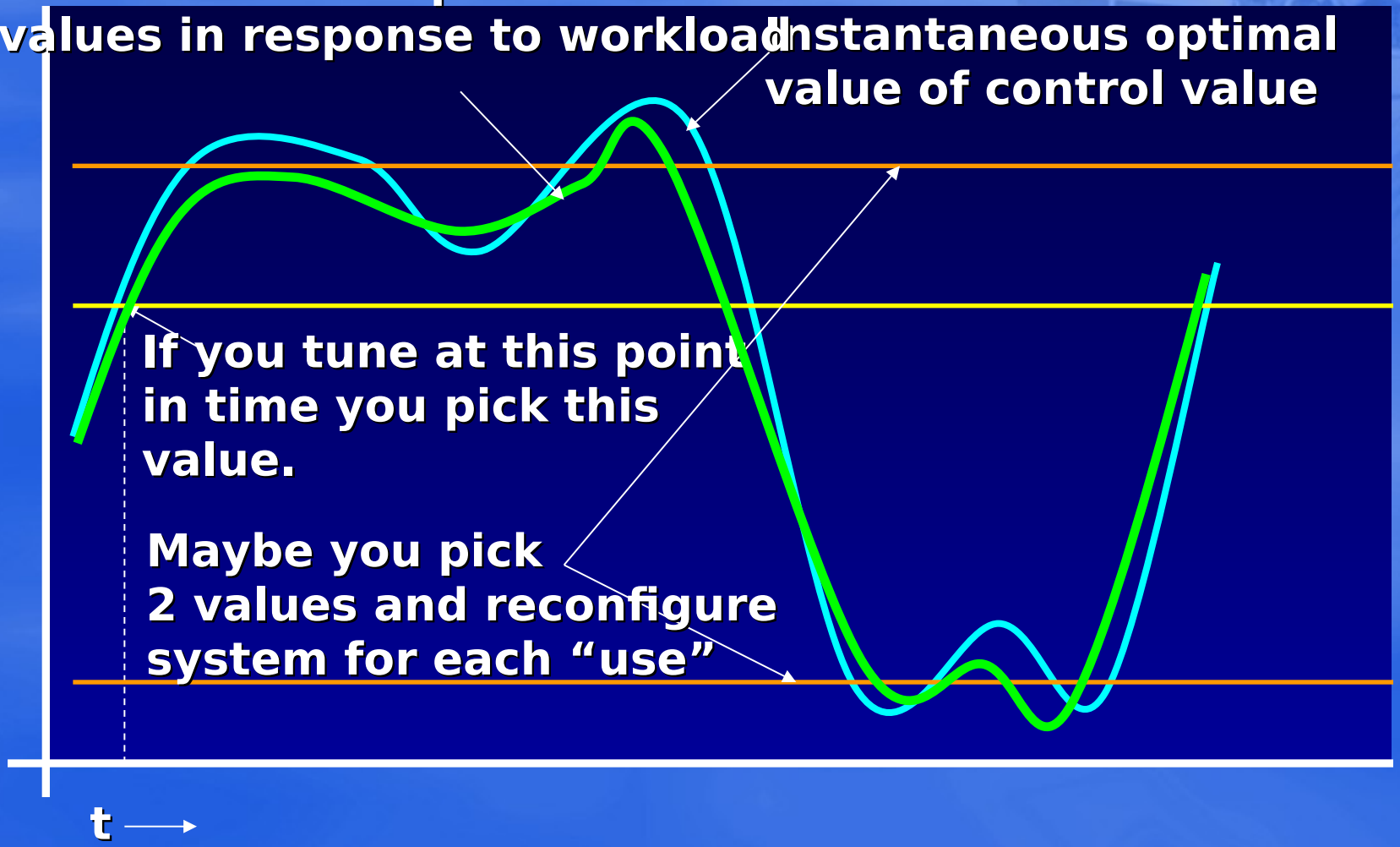
- Performance
- Ease of use
- SAR - ilities
  - Scalability
  - Availability
  - Reliability

Ease Of Use

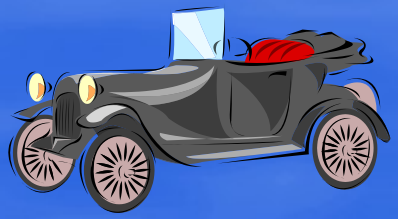
- Key Theme –
  "Simple but NOT simplistic"
- Static configuration parameters replaced with dynamic algorithms employing adaptive feedback
- Administrative control retained to manage system wide resources
  - I.e., You can still constrain the amount of memory used by SQL Server – if you want

Dynamic Control

Dynamic control algorithms maintain "near optimal" values in response to workload



Dynamic Control Analogy

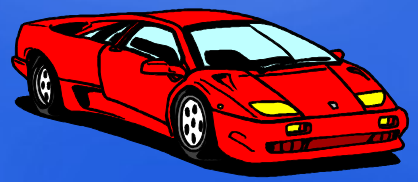


Ignition Timing user controlled by lever
- Classic “knob”

SQL Server 6.x
Oracle
Most current DBs

Vacuum Advance
- Uses simple feedback
- Major advance
- Great for masses

SQL Server 7.0 – SQL Server 2000



Full Feedback:
- Continuously adjusted
- Monitors: temp, speed, engine response, etc.

SQL Server 2000

Ease Of Use – Admin

- Instrumented SQL Server and collected data from Microsoft® Data Center
- Collected statistics on SQL Server restarts
- The top restart reasons?
  - #1: Cycle server to kick users out.
  - #2: Cycle server due KILL <spid> not completing

Ease Of Use - Admin

- Ability to kill users in a particular database
  - E.g., ALTER DATABASE <dbname>
  - SET SINGLE_USER WITH ROLLBACK
- All database state changes made through new Alter DB syntax

Ease Of Use - Admin

- KILL <spid> improvements
  - Administrators would KILL spid in response to runaway transaction
  - KILL would be rolling back large transaction
  - Admin. would cycle server to kill the KILL command
  - (Restart recovery then had to roll back transaction that initial KILL command was working on…)
  - KILL command now reports completion status of rollback. i.e. % rollback completion

Ease Of Use - Recovery

- Declarative logging model
  - “Truncate log”/ “select into” are gone…
  - Three recovery modes:
    - Simple
    - Bulk logged
    - Normal
  - Everything “works” with all models i.e., You can “Select … Into” with any model

Simple Recovery Mode

- Equivalent to “truncate log”
- No media recovery
- Enough transaction log kept online
  to ensure restart recovery in the event of system failure

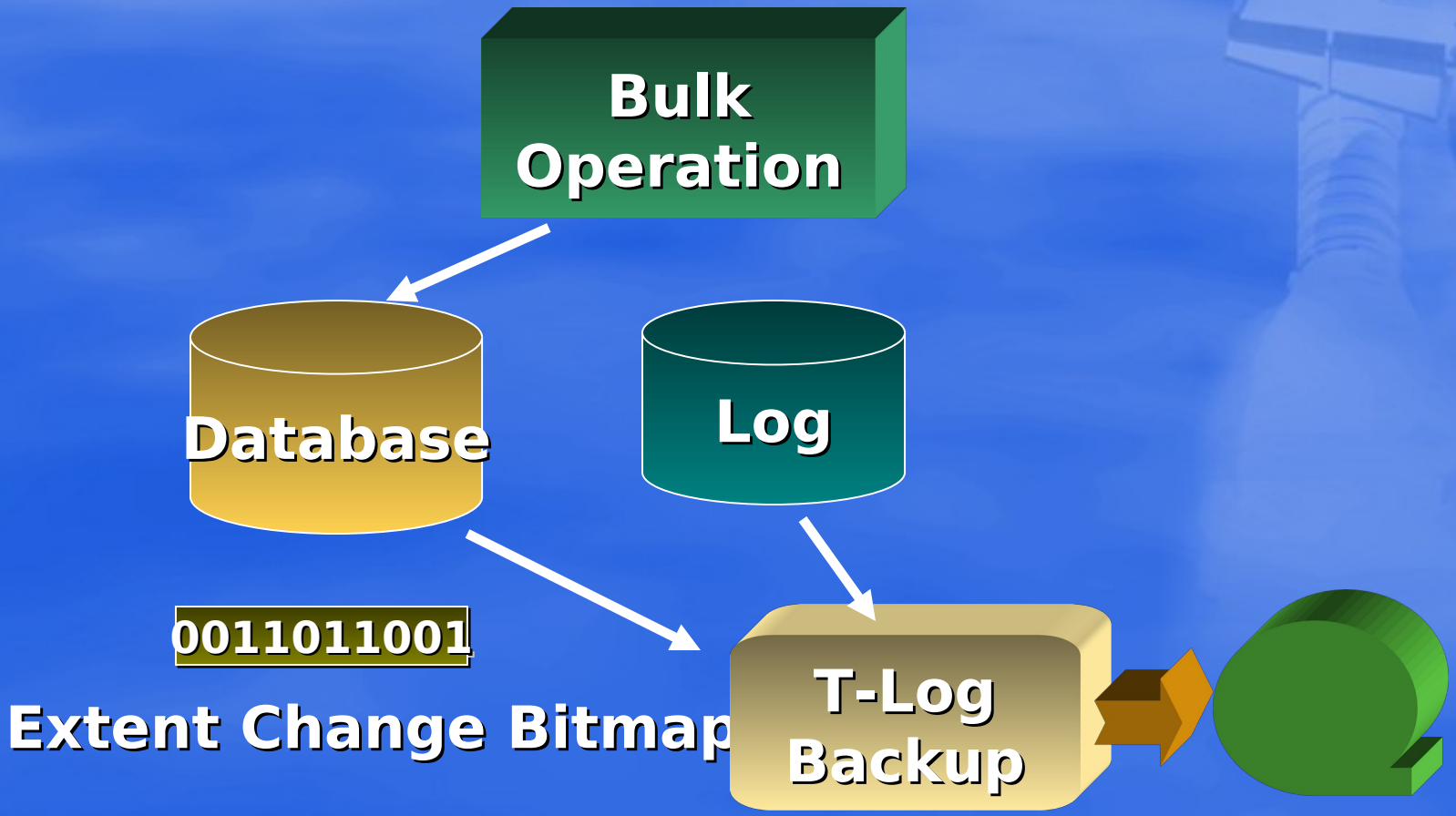
Normal Recovery Mode

- Fully logged
- Database recoverability to any point in time
- Bulk operations use efficient logging (log entire page versus row at a time)
- Provides most protection

Bulk Logged Recovery

- Bulk operations: select into, BCP, WRITETEXT are not logged
- Set of extents touched by all bulk operations is recorded
- Transaction log backup includes contents of extents involved in bulk operation
- (Avoids “staging” bulk data in online log)

Bulk Logged Recovery Mode



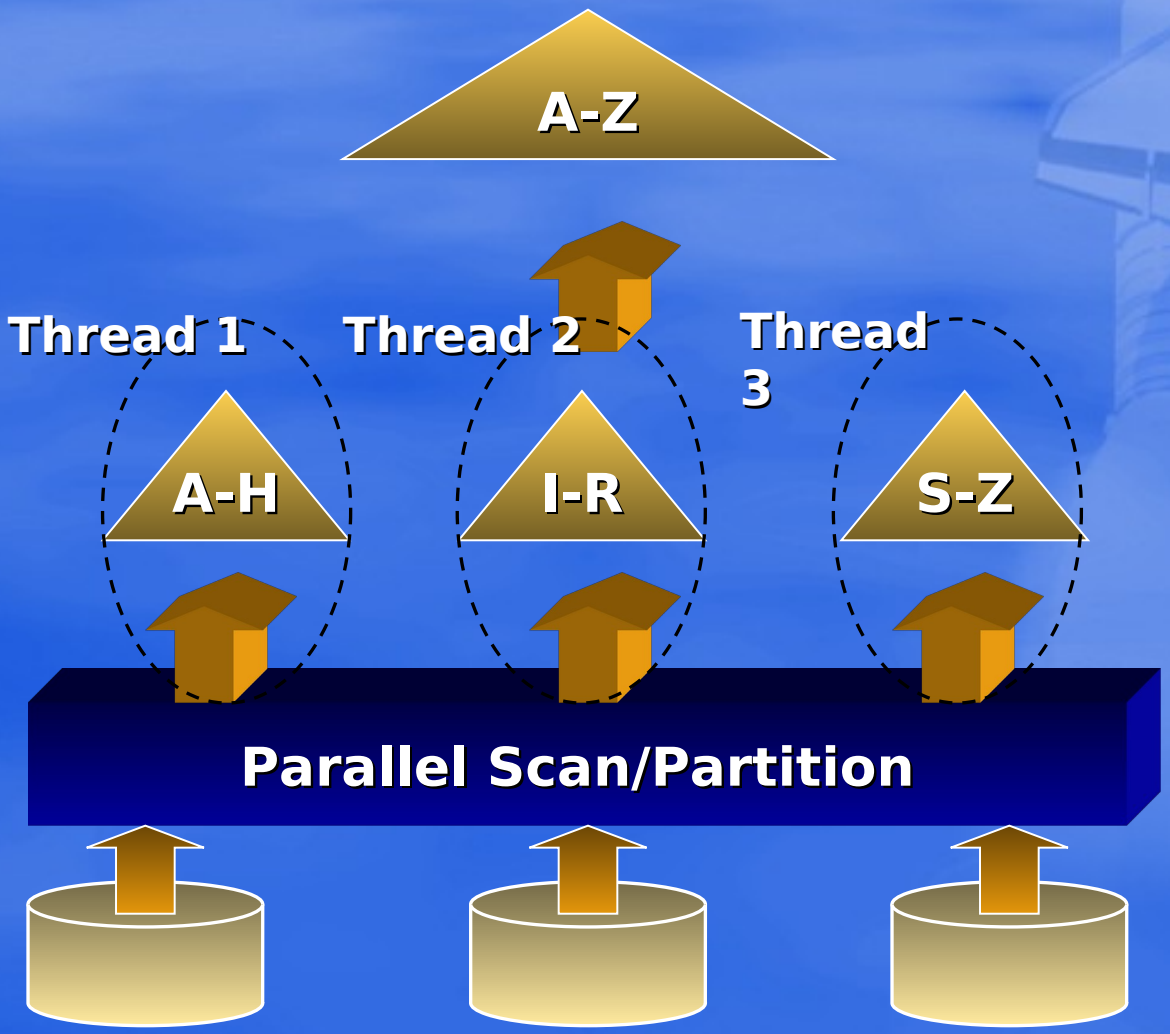
SQL Server 2000 SE Features

- Performance
- Ease of use
- SAR - ilities
  - Scalability
  - Availability
  - Reliability

Parallel Index Build

- Parallel Index Build
- Partitioned sub-trees built by multiple threads
  - Each parallel thread gets a distinct range of values
- Parallel threads fed by fast parallel scan
- Sub-trees “stitched” together in final step

Parallel Index Build



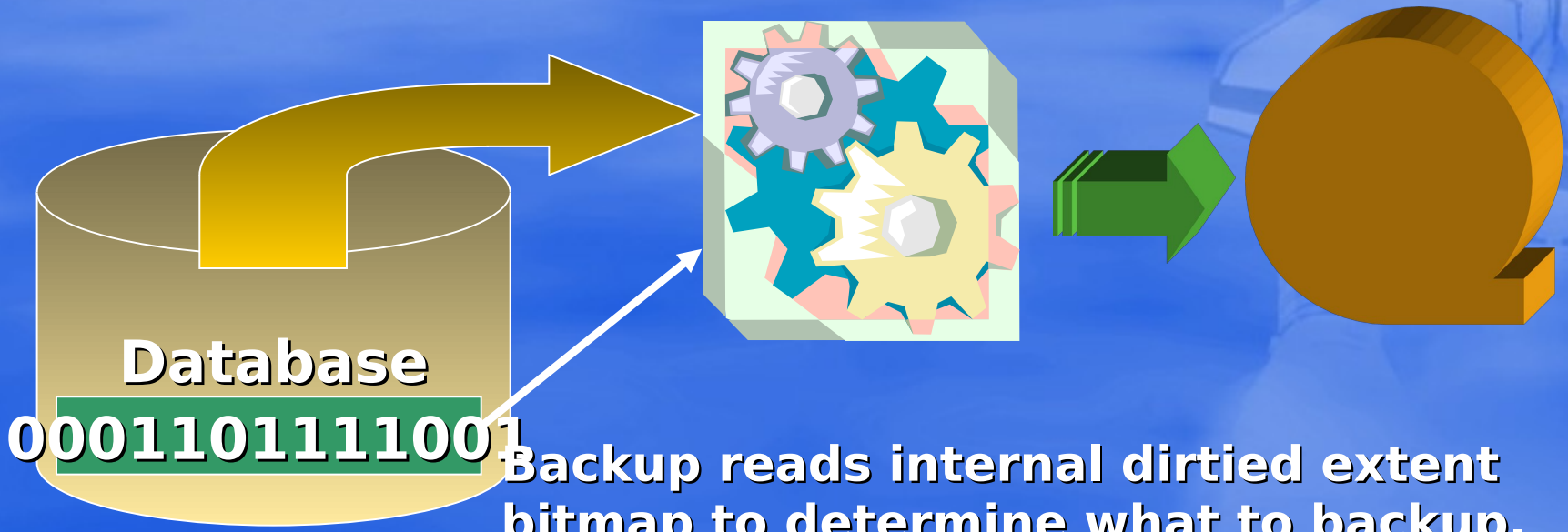
Backup Restore Features

- Differential Backup
- Recover to Mark
- Fast fail back from Standby to Primary
- “Snapshot” Backup/Recovery
- Improved recovery from isolated application or operator errors
- Improved security for backups
- Other enhancements

Feature – Fast Differential Backup

- **Differential backup**
  - **Each extent modified since previous full backup marked in a bitmap**
  - **Differential backup consults bitmap and only backs up changes**
  - **Bitmap provides I/O map to drive skip-sequential I/O**

Differential Backup

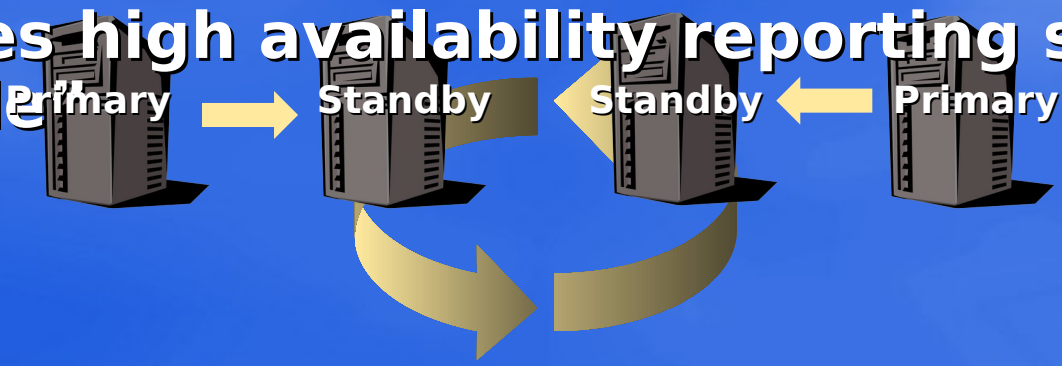


Backup reads internal dirtied extent bitmap to determine what to backup.

Differential Backup time scales with amount of dirtied data rather than the size of the database.

Log Shipping
Fast Failback

- Database restore not required if the data and log files are not damaged
  - BACKUP LOG … TO … WITH { NORECOVERY | STANDBY }
- Log backups taken on secondary can be applied back on primary
- Very useful for planned failover.
- Enables high availability reporting server “toggle”



Snapshot Backup/Restore

- Functionally equivalent to full database
  or file/filegroup backups
  - Roll forward using conventional differential and log backups
    - Can be used to initialize standby database
  - History maintained in MSDB
- Third party VDI* application with storage system support (split mirror or copy-on-write)

Online Index Reorganization

- Reorgs index, online!
- Realigns pages in logical key order
- Moves rows to reestablish desired fill factor
- Fully logged and recovered
- Improved tool to analyze and report fragmentation
- Needed only to reduce disk seeks when doing an ordered index scan

Online Index Reorganization

SQL Server 2000 SE Features

- Performance
- Ease of use
- SAR – ilities
  - Scalability
  - Availability
  - Reliability
- Other Features

Locking Changes

- Generalized Deadlock Detection
  - Now detects deadlocks across: Locks, Threads, Memory, etc.
- Concurrency improvements
  - Minor cleanup to reduce overlocking, reduce deadlocks
- Explicit “XLOCK” mode hint
  - SELECT * FROM FOO WITH (XLOCK)

Locking Changes

- **Application Locks**
  - **Found many apps creating own locking logic using temp tables**
  - **Allows locking application defined resources**
  - **Transaction or Session scoped**
  - **Example:**

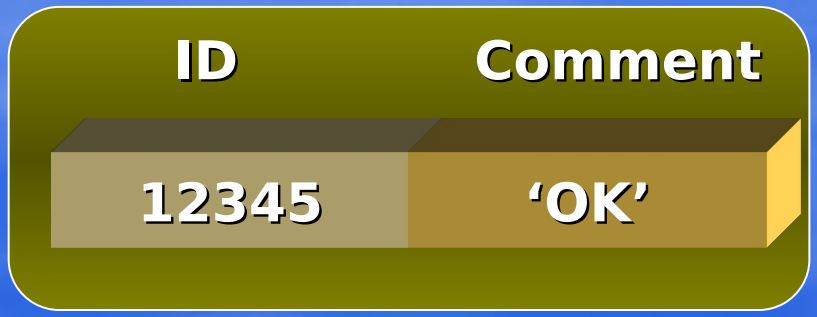
```
USE Northwind
DECLARE @result int
EXEC @result = sp_getapplock
     @Resource =‘Form1’,
     @LockMode = ‘Shared’
```

Feature: In-Row Text

- Allows text/image data to be stored
  in the row with the rest of data
  - Turns two logical I/O's into one
- User settable size threshold
- Need to balance storage density versus I/O win
  - If you very rarely access the text column, you probably shouldn't put it in row since
    it will reduce storage density and scan efficiency

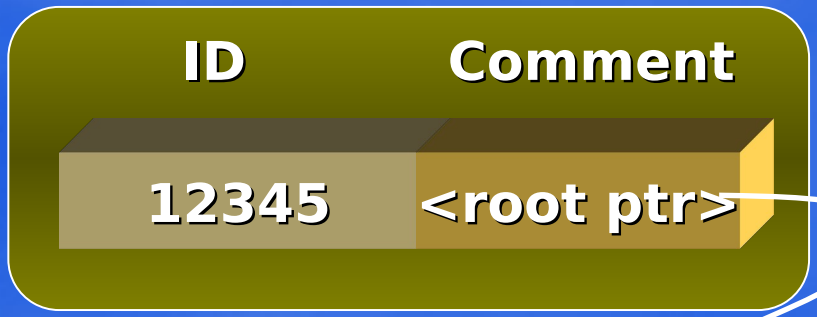
Feature: In-Row Text

Page 11



UPDATE Comments SET COMMENT = 'Well, not so bad but…'
WHERE ID = 12345

Page 11



Page 33

Feature – Expression Push

- Description:
  - Rather than extracting each row from the Storage Engine and applying a filter, aggregate, etc., push the code down into the SE and have the SE apply it against each row
  - Optimizes call path and page setup costs per row
  - Lots of other systems do this…

Feature - Logical Log Marks

- Allows application to place mark in transaction log
- Point in time recovery can reference logical mark name instead of wall clock time
- Recovery can include or stop right before transaction that contains mark
- Typically used with big batch transactions

Logical Log Marks

- Assume you have a purge program that should be run at end quarter. Someone runs it at the wrong time…

1. RESTORE
2. RESTORE LOG … WITH STOPBEFOREMARK = ‘EOQ_PURGE’

Miscellaneous Features

- Descending Index support
  - CREATE INDEX I1 on T1 (C1 ASCENDING, C2 DESCENDING)
- SHRINK LOG … DOES
  - Either shrinks immediately or tells you what to do, i.e., BACKUP LOG
- BACKUP LOG W/ NO_TRUNCATE
  - Works even if primary data file is not available

POWER UP

Microsoft